# CATALOG-BASED REPRESENTATION OF 2D TRIANGULATIONS*

Luca Castelli Aleardi † Olivier Devillers ‡
Abdelkrim Mebarki §

**Abstract**

Several Representations and Coding schemes have been proposed to represent efficiently 2D triangulations. In this paper we propose a new practical approach to reduce the main memory space needed to represent an arbitrary triangulation, while maintaining constant time for some basic queries. This work focuses on the connectivity information of the triangulation, rather than the geometric information (vertex coordinates), since the combinatorial data represents the main part of the storage. The main idea is to gather triangles into patches, to reduce the number of pointers by eliminating the internal pointers in the patches and reducing the multiple references to vertices. To accomplish this, we define and use stable catalogs of patches that are closed under basic standard update operations such as insertion and deletion of vertices, and edge flips. We present some bounds and results concerning special catalogs, and some experimental results that exhibits the practical gain of such methods.

## 1 Introduction

The triangulation is the basic data structure in a large spectrum of application domains, ranging from geometric modeling, to finite elements and interpolation schemes. This data structure has been widely studied from different points of view, and several schemes have been recently proposed for representing triangular meshes. One can use classical Half-Edge-Based representation[10], in which the triangulation is perceived as a set of half-edges. Each half-edge is represented at least with one of its incident vertices, the opposite half-edge, and the previous (or the next) half-edge in the same incident face. Moreover, each vertex stores a reference to an incident half-edge, which yields a global storage cost of $19n$ references for a triangulation of $n$ vertices. In the Face-Based representation[3], the key objects represented in the triangulation are its faces (triangles). Each face maintains references to its three vertices, and to its

*This work has been supported by the french "Aci Masse De Données" Program, via the Geocomp Project, http://www.lix.polythechnique.fr/∼schaeffe/GeoComp

†Laboratoire d'Informatique, École Polytechnique, 91128 Palaiseau cedex, France. amturing@lix.polytechnique.fr

‡INRIA, BP 93, 06902 Sophia Antipolis cedex, France. Olivier.Devillers@inria.fr

§INRIA, BP 93, 06902 Sophia Antipolis cedex, France. abdelkrim.mebarki@gmail.com. Currently Oran University, Algeria

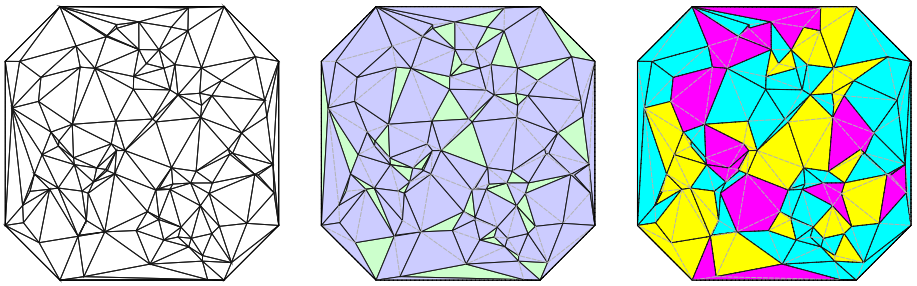

Figure 1: Representing triangulations using stable catalogs gathers triangles into patches to reduce multiple references : leftmost: The basic triangulation ; middle : The same triangulation coded using Quad-Triangle catalog ; rightmost : The same triangulation coded using 3-patches catalog.

three neighbors. In addition, each vertex maintains a reference of an incident face, hence representing a triangulation with $n$ vertices requires $13n$ references. These structures allow an efficient navigation over the triangulation: standard adjacency queries (visiting neighbors), and incident queries (testing incidence between faces, edges and vertices), are all supported in $O(1)$ time.

From the encoding point of view, many solutions have been developed for compression purpose, mainly for triangular meshes. In this case, the triangulation is just implicitly encoded (hence there is no actual data structure), and there does not exist an efficient way to access to the stored data without uncompressing the whole code. From the information theory point of view, we know that representing an arbitrary planar triangulation requires 3.24 *bpv* (bits per vertex), and a linear time optimal encoding has been recently introduced[13] by Poulalhon and Schaeffer. On the practical side, several efficient compression schemes have been proposed, achieving very interesting bit rates, especially in the case of regular meshes[1]. Beyond the usual representation schemes (Half-Edge based and Face based), some compact representations were proposed to reduce the memory cost. Star vertices[9] is proposed by Kallmann and Thalmann. It is a Vertex-Based representation: each vertex handles a list of all of its adjacent vertices (the vertex stores the size of this list), resulting in $6n$ references plus $n$ integers (sizes of lists) to represent the whole triangulation. However, the internal structure no longer has an explicit representation of faces, and queries cost time is proportional to the degree of the involved vertex. In a recent work [7], Gurung and Rossignac recently propose a new compact data structure for triangle meshes requiring $6n$ references: their approach is based on the *Corner Table* representation and exploits a reordering of the triangles. This makes their structure essentially static and, moreover, the access operator to vertices requires more than constant time (the time is proportional to the degree of a vertex). Blandford *et al.*[2] proposed a compact data structure for representing simplicial meshes, requiring in practice 40 *bpf* (bits per face). The representation does admit an Edge-based or Vertex-based representation, providing basic update operations and standard local navigation between triangles (performing these operations takes $O(1)$ time for the case of meshes with bounded vertex degree). To gain in memory, difference vertex labels are used

instead of real pointers, and a preprocessing step consisting of relabelling vertices, for reducing the differences, is needed. This approach takes advantage of properties of graphs with small separators and require some assumptions on the input data.

### Paper's contribution

In previous papers, we have proposed an optimal way of representing a triangulation of $n$ points using 3.24 $n$ bits[5, 6], with an additional storage which is asymptotically negligible (in the case of a triangulation of a topological sphere; for the triangulation bounded by a polygon of arbitrary size the cost is 2.17 bits per triangle). The idea is to gather triangles in tiny patches of size between $\frac{\log n}{12}$ and $\frac{\log n}{4}$, and to introduce a graph of patches to describe adjacency relations between them. Each patch is then represented by a reference to a catalog, consisting of all different tiny patches of size less than $\frac{\log n}{4}$. The whole size of all references to the catalog gives the dominant term of 3.24 *bpv*, while the representation of the graph of patches requires a negligible amount of space. Unfortunately this negligible term is of the form $O\left(n\frac{\log\log n}{\log n}\right)$ with some non-negligible constant which makes the approach essentially of theoretical interest. Nevertheless, the general idea is interesting and can be used in practice with some simplifications and this is the object of the present paper, which presents some *non-asymptotical* results based on the two following remarks:

1– Even if not negligible, the incidence graph of patches is still smaller than the incidence graph of original triangles, allowing to reduce the memory requirements.

2– $\frac{\log n}{12}$ is very small ($\frac{\log 70billions}{12} = 4$) it would be interesting to study in detail the composition of small fixed catalogs suitable for our purpose.

In the sequel we propose some catalogs and evaluate in detail the amount of storage needed for representing triangulations using this approach. The implementation shows that the expected improvements are actually obtained in practice.

## 2 Definitions

A *Catalog* $\mathcal{C}$ is a collection $\{t_1, \ldots, t_p\}$ of planar triangulations with a simple boundary of arbitrary size, called *patches* (the $t_i$ are called *tiny* triangulations in previous papers[5, 6]). A *stable catalog* for a given operation is a catalog, where the result of each update operation applied to a triangulation formed by one or several patches of the catalog is either included in the catalog, or decomposable (within a restricted neighborhood) to some elements of the catalog. The interest of such a catalog is that a triangulation $\mathcal{T}$ could be represented with a disjoint union of patches in $\mathcal{C}$: $\mathcal{T} = \bigcup_{j\in J} t_{j_i}$ $(1 \leq j_i \leq p)$. Examples of stable catalogs are shown in Figure 2. A catalog $\mathcal{C}$ is *minimal* if no patch $t_i$ can be obtained as the disjoint union of other patches in $\mathcal{C}$.

In this paper, we aim to support the insertion/deletion of degree-3 vertices and the edge flip as update operations.

Now one may ask, given a parameter $k$, how to find a minimal catalog $\mathcal{C}$ whose patches have each at least $k$ triangles. We may proceed as follows:

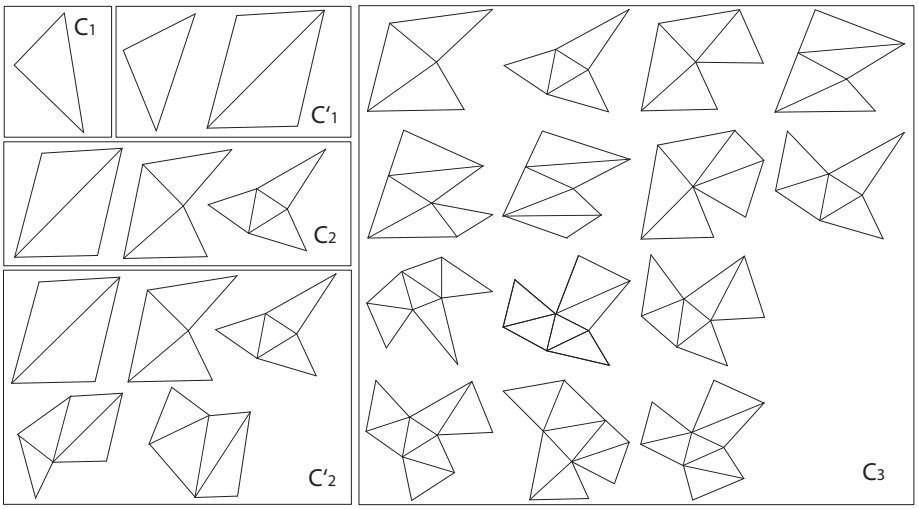


Figure 2: Five stable catalogs for the elementary operations: insertion and deletion of degree-3 vertex and edge flip. (1) The trivial catalog with only one triangle $\mathcal{C}_1$ (2) Triangle-Quad catalog $\mathcal{C}'_1$ (3) The minimal catalog with at least 2 triangles/patch $\mathcal{C}_2$ (4) A non-minimal catalog with at least 2 triangles/patche $\mathcal{C}'_2$ (5) The minimal catalog with at least 3 triangles/patch $\mathcal{C}_3$.

Firstly, the catalog should contain all of the patches with $k$ triangles. Also, it includes all of the combinations with between $k+1$ and $2k-1$ triangles, since there is no way to represent a triangulation containing less than $2k-1$ triangles with only patches of $k$ triangles. Then, we have to investigate all of the configurations produced when applying an update operation on the patches of the catalog. Whenever we obtain a configuration that is not decomposable into elements of the catalog $\mathcal{C}$, we add it to $\mathcal{C}$.

# 3 Simple Catalogs

In this section we present some simple catalogs, provided with upper bounds on the memory requirements of our structure, based on basic combinatorial assumptions. The first trivial catalog $\mathcal{C}_1$ is just composed of one triangulation having a single triangle and yields to the usual representation using $13n$ references.

## 3.1 The Triangle-quad catalog

The first non-trivial catalog $\mathcal{C}'_1$ is composed of two basic elements: triangles and quadrangles. It is clear that $\mathcal{C}'_1$ is stable for the update operations considered above, but not minimal.

The triangulation is then represented by two sets, one storing the list of triangles, and one for the quads. The triangle representation remains unchanged: 6 references are required (3 for vertices and 3 for neighbors). For each quadrangle only 4 references to vertices and 4 to neighbors are needed, which makes save 2 references over each triangle converted into a quadrangle. The way of a

quad is triangulated may be implicitly represented by numbering the vertices in the quad, i.e. by convention the quad diagonal always join vertices 1 and 3.

The gain in space is then proportional to the number of constructed quadrangles. The maximum we can obtain is $9n$ references for a triangulation of $n$ points (for a quadrangulation). However, it is not always feasible to construct a triangulation only with quadrangles.

### 3.1.1 Static representation

In the static case we assume that the triangulation is entirely constructed: it only remains to get a partition into patches (as triangles and quads), as required by our scheme. Several approaches have been proposed to convert triangulations to quadrangulations[8, 14], or create a quadrangulation from an input set of points[15]. In general, there is no obvious way to guarantee that an arbitrary triangulation could be converted to a quadrangulation. This is even impossible in some cases (when the size of the boundary is odd for example[4]). Nevertheless, it is always possible to convert a triangulation of a topological sphere to a quadrangulation, since Petersen's theorem[12] guarantees that each 3-regular bridgeless connected graph has a complete matching. Using this theorem on the dual graph gives that all triangles can be gathered in quadrangles.

### 3.1.2 Dynamic representation

In the dynamic case we allow update operations on the triangulation, namely vertex insertion/deletion and edge flip. To keep the update time constant, we require that the subdivision of patches is decided locally which prevents us from using the static result.

**Lemma 1** *Let be $\mathcal{T}$ a planar triangulation with $n$ vertices. Then an explicit Face-based representation using triangles and quads requiring less than $10.6n$ references can be maintained dynamically.*

**Proof.** We may assume we are given a decomposition of $\mathcal{T}$ such that there is no pair of adjacent triangles (since adjacent triangles can be gathered to form a quadrangle). This property can easily be maintained under the update operations, indeed after each update, we verify locally the neighbors of each involved triangle and if another triangle is found we create a quad (more details can be found in [11]).

Let $t$ and $q$ denote respectively the number of triangles and quadrangles in the decomposition of $\mathcal{T}$, and $b$ denote the number of edges on the boundary. The total number of original triangles in $\mathcal{T}$ is

$$2n - b - 2 = t + 2q. \tag{1}$$

The number edges of $\mathcal{T}$ is known to be $3n - 3 - b$ so

$$3n - b - 3 = e_{int} + e_{tr/quad} + e_{quad/quad} \tag{2}$$

where $e_{int}$ is the number of edges internal to a quadrangle (diagonals), $e_{tr/quad}$ is the number of edges shared by quadrangles and triangles, and $e_{quad/quad}$ is

the number of edges shared by two quads; since triangles are not adjacent, there is no other kind of edges. We have $e_{int} = q$ and $e_{tr/quad} = 3t$, thus

$$e_{quad/quad} = 3n - 3t - q - 3 - b \geq 0. \tag{3}$$

Using 1 and 3, we get the following:

$$q > \frac{3}{5}n - \frac{2}{5}b + \frac{3}{5}. \tag{4}$$

That means that we have at most $\frac{4}{5}n - \frac{1}{5}b - \frac{6}{5}$ triangles. When the boundary size is small compared to $n$, the triangulation can be represented with about $\frac{53}{5}n = 10.6n$ references, instead of $13n$ in the basic representation, inducing a saving of 19%. □

#### 3.1.3 On the size of references

In the classical representation there are $2n$ triangles and $n$ vertices; hence, a vertex reference needs $\log n$ bits and a triangle reference $1 + \log n$ bits. For neighboring relation between triangles, we may want to add to the reference an index in $\{0, 1, 2\}$ to be able to find easily the reciprocal references; this yields to an $3 + \log n$ bits cost per reference. For a triangulation represented with the triangle/quad catalog described above, we have $t \leq \frac{4}{5}n$ and $q \leq n$; hence, a triangle or quad reference use $\log n$ bits, but we must add to it one bit to decide if it is a triangle or a quad, and 2 bits for the reciprocal references as previously. All together a face reference cost $3 + \log n$ bits in both cases.

### 3.2 Catalog with at least 2-triangles per patch

#### 3.2.1 Stable catalog

First, we consider the minimal catalog $\mathcal{C}_2$ with at least 2 triangles per patch. This catalog is drawn in Fig. 2. In a representation using $\mathcal{C}_2$, it is plain to observe that the number of patches is at most $n$ and that we save at least two references per triangle going from the usual $13n$ references to $9n$ references and saving 31% of the memory.

**Lemma 2** *Let be $\mathcal{T}$ a planar triangulation with $n$ vertices. Then an explicit Face-based representation using the minimal stable catalog $C_2$ requiring less than $8.5n$ references can be maintained dynamically.*

**Proof.** We may assume we are given a decomposition of $\mathcal{T}$ such that there is no quadrangle adjacent to more than two other quadrangles.

In fact, if a quadrangle is adjacent to three other quadrangles, two of them have to be incident to a vertex reached by a diagonal (Fig 3). Hence, we can construct a pentagon using these quadrangles.

This property can be easily maintained under the update operations. Indeed, after each update, we verify locally the neighbors of each involved quadrangle and construct the maximum number of pentagons we can do.

Let $q$, $p$ and $h$ denote respectively the number of quadrangles, pentagons, and hexagons in the decomposition of $\mathcal{T}$, and $b$ denote the number of edges on

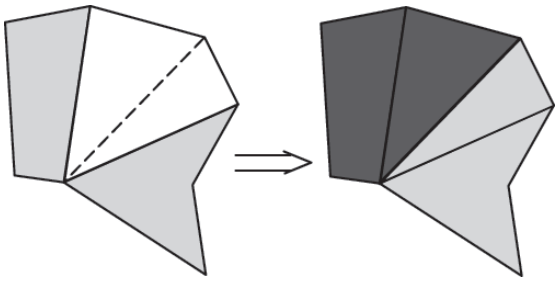

Figure 3: Pentagons can be constructed when a quadrangle has two adjacent quadrangles incident to a vertex reached by a diagonal.

the boundary. To obtain the worst storage case, we consider that there is no hexagon in the triangulation. The total number of triangles is

$$2n - b - 2 = 2q + 3p. \tag{5}$$

The number of edges is known to be $3n$, and is equal to

$$3n - b - 3 = e_{int} + e_{quad/quad} + e_{rem}, \tag{6}$$

where $e_{int}$ is the number of edges internal to the quadrangles and pentagons, (diagonals) ; and $e_{rem}$ is the sum of the edges shared by the pentagons and those between pentagons and quads.

Since there are not three quadrangles adjacent to the same quadrangle, $e_{quad/quad}$ is bounded by $2q/2$, and hence $e_{rem}$ is at least $2q$, thus :

$$e_{rem} = 3n - 2q - 2p - b - 3 \geq 2q. \tag{7}$$

Using 5 and 7, we get

$$p > \frac{1}{4}n - \frac{1}{4}b - \frac{1}{4}. \tag{8}$$

That means that we have at most $\frac{5}{8}n - \frac{7}{8}b - \frac{5}{8}$ quadrangles. The triangulation can be represented with $\frac{17}{2}n = 8.5n$ references, instead of $13n$ in the basic representation, inducing a saving of 35%. □

### 3.2.2 Other catalog

An alternative is to use catalog $\mathcal{C}'_2$ described on Fig. 2 which is not minimal. In a way similar to what we have done with $\mathcal{C}_1$, we can require that two quads are not adjacent. This is feasible since any two adjacent quadrangles can be converted into a hexagon. In this case (assuming that there is no hexagons in the worst case) we cannot get more than $\frac{5}{11}n$ quadrangles. On the other hand, we have at least $\frac{4}{11}n$ pentagons, which yields, in the worst case, a storage cost of $\frac{91}{11}n = 8.27n$, corresponding to a gain of 36% over the basic representation (using the same counting argument as before).

### 3.2.3 On the size of references

For $\mathcal{C}'_2$ there is less than $\frac{1}{2}n$ faces of each kind, and thus $-1 + \log n$ bits to refer to a face. We need some additional bits to distinguish the kind of the neighbor

(quad, pentagon or hexagon) and the index of the edge of it. Thus we have $4+5+6=15$ cases which can be distinguished with 3 bits yieldings again to a face reference of $3+\log n$ bits. For $\mathcal{C}_2$ the number of quads is less under control and we need to use $4+\log n$ bits per face reference.

### 3.3 Minimal catalog with at least 3-triangles per package

Fig. 2 shows the minimal stable catalog $\mathcal{C}_3$ for the update operations with no less than three triangles per patch. This catalog contains triangulations having between 3 and 7 triangles, whose boundary has size between 5 and 9. Heptagons, octagons and enneagons are represented respectively with 14, 20 and 24 references. This produces respectively gains of $\frac{16}{5}$, $\frac{10}{3}$ and $\frac{24}{7}$ over each triangle converted into one of these patches. The worst case cost for this catalog occurs when all the patches are pentagons. In this case, the global storage cost of the triangulation is $\frac{23}{3}n = 7.67n$ references, which is equivalent to 41% gain in memory space over the basic representation.

## 4 Experimental Results

We have implemented the trivial catalog $\mathcal{C}_1$, the quad-triangle catalog $\mathcal{C}'_1$, and the minimal catalog including patches having at least 2 triangles per patch $\mathcal{C}_2$. The experimental results shown in Table 1 are obtained by computing the Delaunay triangulation, from uniform random distributions of points, adopting an incremental algorithm. The triangulation is built incrementally, and the patches are created in the same way. The Delaunay property is maintained by propagating flips from the new inserted point. An optimization step is performed after each insertion operation for the two catalogs $\mathcal{C}'_1$ and $\mathcal{C}_2$: We look for adjacent triangles to be converted into quadrangles for $\mathcal{C}_1$, and for quadrangle strips that could be converted into pentagons or hexagons, in order to reduce the number of quadrangles and pentagon strips and thus to maximize the number of hexagons, for $\mathcal{C}_2$.

In term of software engineering, the use of catalogs to code triangulations requires some intermediate levels between the storage level (containing the real objects of the catalog which are elements of the catalog) and the user level (in which only logical faces appear). In the case of $C_2$ catalog, the Triangulation is a quadruple container: the first one is for vertices; the second one is for quadrangles; the third one is for pentagons; and the fourth one is for hexagons. At a higher level, the user manipulates *faces*, that are triangles, without worrying about the internal representation. These details affect the time processing of the triangulation in both steps: construction and manipulation (more details are given in [11]).

## 5 Conclusion

In this paper, we have proposed a new representation for triangulations inspired from previous theoretical work[6]. We describe several catalogs of tiny patches that are stable under relevant update operations which can be used as patterns to recognize in the triangulation. This allows us to avoid representing the details of the connectivity inside a patch (since they are common to all patches of

| 10M of points (20M of triangles) | | | | | |
|---|---|---|---|---|---|
| Catalogs | $C_1$ | $C'_1$ | $C'_1$ with optimization | $C_2$ | $C_2$ with optimization |
| Triangles | 100% | 40% | 14% | - | - |
| Quadrangles | - | 60% | 86% | 60% | 24% |
| Pentagons | - | - | - | 32% | 35% |
| Hexagons | - | - | - | 8% | 41% |
| Minimal connectivity saving(%) | - | - | 19% | - | 35% |
| Effective connectivity saving(%) | - | 19% | 27% | 35% | 40% |
| Memory | 578 Mb | 480 Mb | 442 Mb | 402 Mb | 376 Mb |
| Total memory saving(%) | - | 17% | 24% | 31% | 35% |
| Timing | 125 s | 382 s | 626 s | 393 s | 939 s |

Table 1: First lines show the percentage of triangles belonging to each kind of triangulation in the catalog. The minimal connectivity saving is the one computed in 3. The effective connectivity saving uses the actual use of the catalog. The total memory saving refers to the memory used by the program.

the same kind, and thus represented only once). The previous work[6] was theoretically optimal but quite far of a practical application since it was very complicated to implement and since the asymptotic optimal behavior is reached for highly unrealistic size of triangulation. This work gives practical analysis and experimental evidence that a simplified version of this work allows one to reduce significantly the storage needed allowing a trade-off between time and space needed by a triangulation.

# References


[1] P. Alliez and C. Gotsman. Recent advances in compression of 3d meshes. In N. Dodgson, M. Floater, and M. S. Springer-Verlag, editors, *Advances in Multiresolution for Geometric Modelling*, pages 3–26. Springer-Verlag, 2005.

[2] D. K. B. G. E. Blelloch D. E. Cardoze C. Kadow. Compact representations of simplicial meshes in two and three dimensions. *Internat. J. Comput. Geom. Appl.*, 15:3–24, 2005.

[3] J.-D. Boissonnat, O. Devillers, S. Pion, M. Teillaud, and M. Yvinec. Triangulations in CGAL. *Comput. Geom. Theory Appl.*, 22:5–19, 2002.

[4] P. Bose and G. Toussaint. Characterizing and efficiently computing quadrangulations of planar point sets. *Comput. Aided Geom. Des.*, 14(8):763-785,1997.

[5] L. Castelli Aleardi, O. Devillers and G. Schaeffer. Succinct representation of triangulations with a boundary. In *Proc. of WADS 2005*, p. 134-145, 2005.

[6] L. Castelli Aleardi, O. Devillers and G. Schaeffer. Succinct representations of planar maps. *Theoretical Computer Science*, 408:174–187, 2008.

[7] T. Gurung and J. Rossignac. SOT: compact representation for tetrahedral meshes. In *Proc. of Symp. on Solid and Physical Modeling*, p. 79-88, 2009.

[8] E. Heighway. A mesh generator for automatically subdividing irregular polygons into quadrilaterals. In *IEEE Transactions on Magnetics*, , 19(6):2535–2538, 1983.

[9] M. Kallmann and D. Thalmann Star-vertices: a compact representation for planar meshes with adjacency information *J. Graph. Tools*, 6(1):7-18, 2001.

[10] L. Kettner. Using generic programming for designing a data structure for polyhedral surfaces. *Computational Geometry – Theory and Applications*, 13:65–90, 1999.

[11] A. Mebarki. *Implantation de structures de données compactes pour les triangulations*. PhD thesis, Université de Nice-Sophia Antipolis, France, 2008.

[12] J. Petersen. Die Theorie der regularen Graphs (The theory of regular graphs). Acta Mathematica, 15:193– 220, 1891.

[13] D. Poulalhon and G. Schaeffer. Optimal coding and sampling of triangulations. *Algorithmica*, 46:505–527, 2006.

[14] S. Ramaswami, P.A. Ramos and G.T. Toussaint. Converting triangulations to quadrangulations. *Comput. Geom. Theory & Applications* 9(4):257-276, 1998.

[15] G. Toussaint. Quadrangulations of planar sets. In *Proc. 4th Workshop Algorithms Data Struct.*, *Lecture Notes Comput. Sci.*, 955:218–227. Springer-Verlag, 1995.